COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco

University of Massachusetts Amherst. Fall 2019.

Lecture 24 (Final Lecture!)

- Problem Set 4 due Sunday 12/15 at 8pm.
- Exam prep materials (including practice problems) posted under the 'Schedule' tab of the course page.
- I will hold office hours on both Tuesday and Wednesday next week from **10am to 12pm** to prep for final.
- SRTI survey is open until 12/22. Your feedback this semester has been very helpful to me, so please fill out the survey!
- `https://owl.umass.edu/partners/courseEvalSurvey/uma/`

Last Class:

- Compressed sensing and sparse recovery.
- Applications to sparse regression, frequent elements problem, sparse Fourier transform.

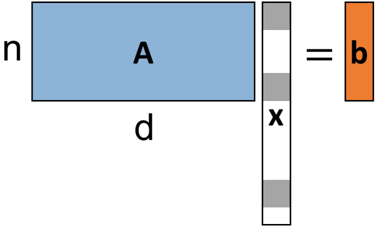
Last Class:

- Compressed sensing and sparse recovery.
- Applications to sparse regression, frequent elements problem, sparse Fourier transform.

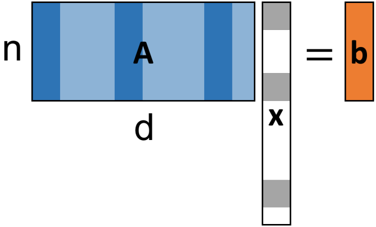
This Class:

- Finish up sparse recovery.
- Solution via basis pursuit. Idea of convex relaxation.
- Wrap up.

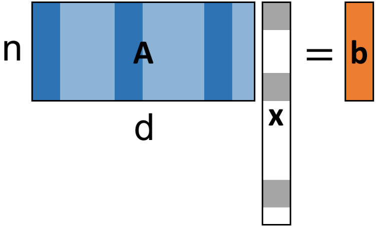
Problem Set Up: Given data matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ with $n < d$ and measurements $\mathbf{b} = \mathbf{A}\mathbf{x}$. Recover $\mathbf{x}$ under the assumption that it is k-sparse, i.e., has at most $k \ll d$ nonzero entries.



Problem Set Up: Given data matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ with $n < d$ and measurements $\mathbf{b} = \mathbf{A}\mathbf{x}$. Recover $\mathbf{x}$ under the assumption that it is k-sparse, i.e., has at most $k \ll d$ nonzero entries.



Problem Set Up: Given data matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ with $n < d$ and measurements $\mathbf{b} = \mathbf{A}\mathbf{x}$. Recover $\mathbf{x}$ under the assumption that it is k-sparse, i.e., has at most $k \ll d$ nonzero entries.

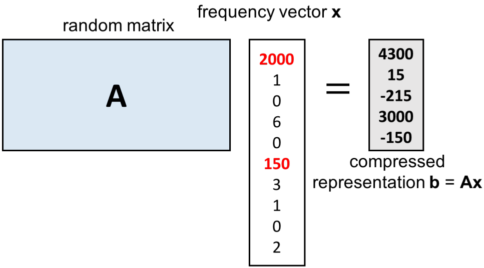


Last Time: Proved this is possible (i.e., the solution $\mathbf{x}$ is unique) when $\mathbf{A}$ has Kruskal rank $\geq 2k$.

$$\mathbf{x} = \underset{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}}{\arg\min} \|\mathbf{z}\|_0,$$

Kruskal rank condition can be satisfied with n as small as $2k$

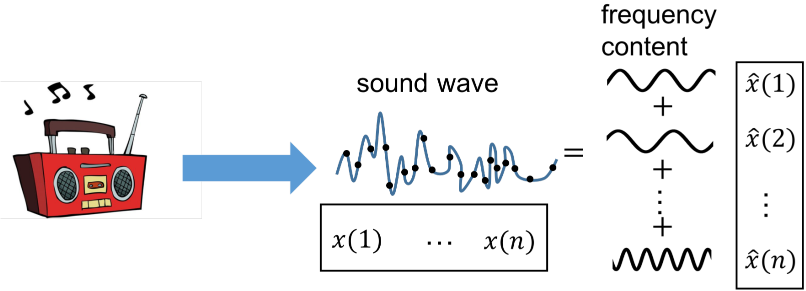
FREQUENT ITEMS COUNTING



- A frequency vector with k out of n very frequent items is approximately k-sparse.
- Can be approximately recovered from its multiplication with a random matrix $\mathbf{A}$ with just $m = \tilde{O}(k)$ rows.
- $\mathbf{b} = \mathbf{A}\mathbf{x}$ can be maintained in a stream using just $O(m)$ space.
- Exactly the set up of Count-min sketch in linear algebraic notation.

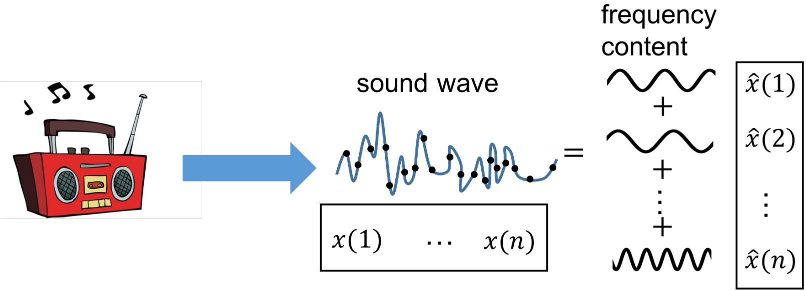
SPARSE FOURIER TRANSFORM

Discrete Fourier Transform: For a discrete signal (aka a vector) $\mathbf{x} \in \mathbb{R}^n$, its discrete Fourier transform is denoted $\widehat{\mathbf{x}} \in \mathbb{C}^n$ and given by $\widehat{\mathbf{x}} = \mathbf{F}\mathbf{x}$, where $\mathbf{F} \in \mathbb{C}^{n \times n}$ is the discrete Fourier transform matrix.



SPARSE FOURIER TRANSFORM

Discrete Fourier Transform: For a discrete signal (aka a vector) $\mathbf{x} \in \mathbb{R}^n$, its discrete Fourier transform is denoted $\widehat{\mathbf{x}} \in \mathbb{C}^n$ and given by $\widehat{\mathbf{x}} = \mathbf{F}\mathbf{x}$, where $\mathbf{F} \in \mathbb{C}^{n \times n}$ is the discrete Fourier transform matrix.

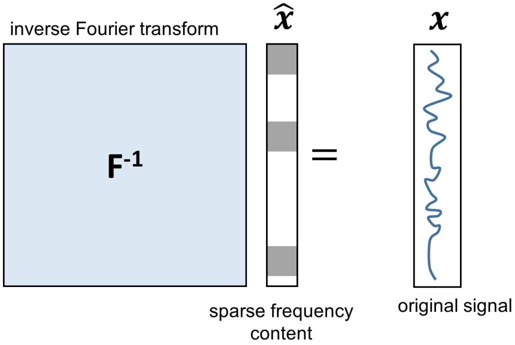


For many natural signals $\widehat{\mathbf{x}}$ is approximately sparse: a few dominant frequencies in a recording, superposition of a few radio transmitters sending at different frequencies, etc.

When the Fourier transform $\widehat{x}$ is sparse, can recover it from few measurements of x using sparse recovery.

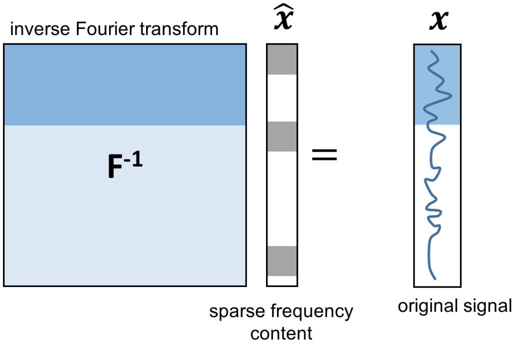
When the Fourier transform $\widehat{\mathbf{x}}$ is sparse, can recover it from few measurements of $\mathbf{x}$ using sparse recovery.

- $\widehat{\mathbf{x}} = \mathbf{F}\mathbf{x}$ and so $\mathbf{x} = \mathbf{F}^{-1}\widehat{\mathbf{x}} = \mathbf{F}^T\widehat{\mathbf{x}}$ ($\mathbf{x}$ = signal, $\widehat{\mathbf{x}}$ = Fourier transform).



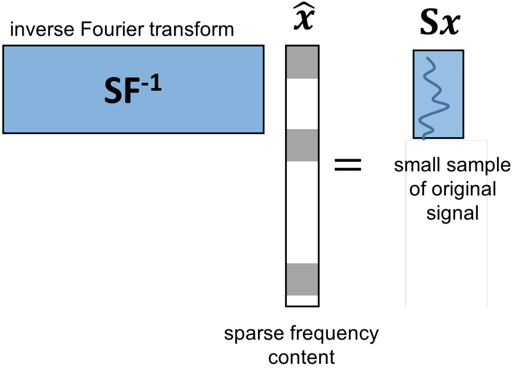
When the Fourier transform $\widehat{\mathbf{x}}$ is sparse, can recover it from few measurements of $\mathbf{x}$ using sparse recovery.

- $\widehat{\mathbf{x}} = \mathbf{F}\mathbf{x}$ and so $\mathbf{x} = \mathbf{F}^{-1}\widehat{\mathbf{x}} = \mathbf{F}^T\widehat{\mathbf{x}}$ ($\mathbf{x}$ = signal, $\widehat{\mathbf{x}}$ = Fourier transform).



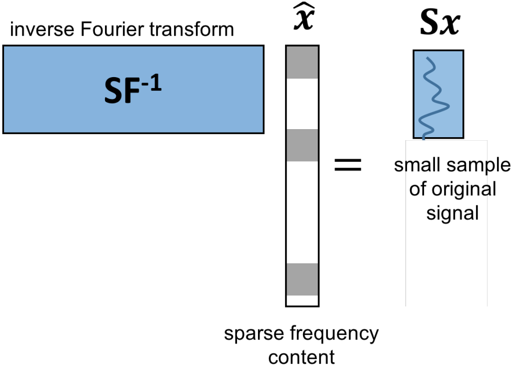
When the Fourier transform $\widehat{x}$ is sparse, can recover it from few measurements of x using sparse recovery.

- $\widehat{x} = Fx$ and so $x = F^{-1}\widehat{x} = F^T\widehat{x}$ (x = signal, $\widehat{x}$ = Fourier transform).



When the Fourier transform $\widehat{\mathbf{x}}$ is sparse, can recover it from few measurements of $\mathbf{x}$ using sparse recovery.

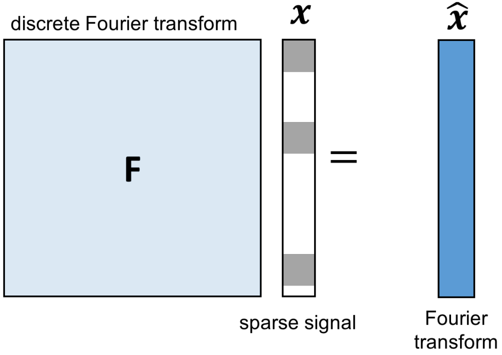
- $\widehat{\mathbf{x}} = \mathbf{F}\mathbf{x}$ and so $\mathbf{x} = \mathbf{F}^{-1}\widehat{\mathbf{x}} = \mathbf{F}^T\widehat{\mathbf{x}}$ ($\mathbf{x}$ = signal, $\widehat{\mathbf{x}}$ = Fourier transform).



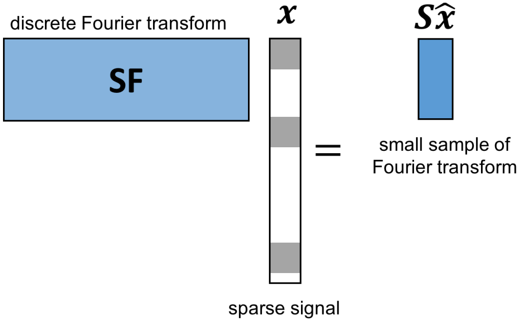
Translates to big savings in acquisition costs and number of sensors.

Other Direction: When x itself is sparse, can recover it from few measurements of the Fourier transform $\widehat{x}$ using sparse recovery.

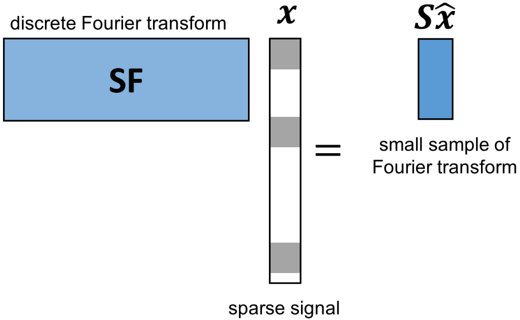
Other Direction: When $\mathbf{x}$ itself is sparse, can recover it from few measurements of the Fourier transform $\widehat{\mathbf{x}}$ using sparse recovery.



Other Direction: When $\mathbf{x}$ itself is sparse, can recover it from few measurements of the Fourier transform $\hat{\mathbf{x}}$ using sparse recovery.



Other Direction: When x itself is sparse, can recover it from few measurements of the Fourier transform $\hat{x}$ using sparse recovery.



How do we access/measure entries of $S\hat{x}$?

- In seismology, x is an image of the earth's crust, and often sparse (e.g., a few locations of oil deposits).
- Want to recover from a few measurements of the Fourier transform $S\widehat{x} = SFx$.

- In seismology, x is an image of the earth's crust, and often sparse (e.g., a few locations of oil deposits).
- Want to recover from a few measurements of the Fourier transform $\mathsf{S}\widehat{\mathsf{x}} = \mathsf{SFx}$.
- To measure entries of $\widehat{\mathsf{x}}$ need to measure the content of different frequencies in a signal x.

- In seismology, x is an image of the earth's crust, and often sparse (e.g., a few locations of oil deposits).
- Want to recover from a few measurements of the Fourier transform $S\hat{x} = SFx$.
- To measure entries of $\hat{x}$ need to measure the content of different frequencies in a signal x.

- Achieved by inducing vibrations of different frequencies with a vibroseis truck, air guns, explosions, etc and recording the response (more complicated in reality...)

Back to Algorithms

We would like to recover k-sparse $\mathbf{x}$ from measurements $\mathbf{b} = \mathbf{A}\mathbf{x}$ by solving the non-convex optimization problem:

$$\mathbf{x} = \underset{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}}{\arg\min} \|\mathbf{z}\|_0$$

Works if $\mathbf{A}$ has Kruskal rank $\geq 2k$, but very hard computationally.

We would like to recover k-sparse $\mathbf{x}$ from measurements $\mathbf{b} = \mathbf{A}\mathbf{x}$ by solving the non-convex optimization problem:

$$\mathbf{x} = \underset{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}}{\arg\min} \|\mathbf{z}\|_0$$

Works if $\mathbf{A}$ has Kruskal rank $\geq 2k$, but very hard computationally.

Convex Relaxation: A very common technique. Just 'relax' the problem to be convex.

We would like to recover k-sparse $\mathbf{x}$ from measurements $\mathbf{b} = \mathbf{A}\mathbf{x}$ by solving the non-convex optimization problem:

$$\mathbf{x} = \underset{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}}{\arg\min} \|\mathbf{z}\|_0$$

Works if $\mathbf{A}$ has Kruskal rank $\geq 2k$, but very hard computationally.

Convex Relaxation: A very common technique. Just 'relax' the problem to be convex.

Basis Pursuit: $\mathbf{x} = \arg\min_{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}} \|\mathbf{z}\|_1$ where $\|\mathbf{z}\|_1 = \sum_{i=1}^d |\mathbf{z}(i)|$.

We would like to recover k-sparse $\mathbf{x}$ from measurements $\mathbf{b} = \mathbf{A}\mathbf{x}$ by solving the non-convex optimization problem:

$$\mathbf{x} = \arg\min_{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}} \|\mathbf{z}\|_0$$

Works if $\mathbf{A}$ has Kruskal rank $\geq 2k$, but very hard computationally.

Convex Relaxation: A very common technique. Just 'relax' the problem to be convex.

Basis Pursuit: $\mathbf{x} = \arg\min_{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}} \|\mathbf{z}\|_1$ where $\|\mathbf{z}\|_1 = \sum_{i=1}^d |\mathbf{z}(i)|$.

What is one algorithm we have learned for solving this problem?

CONVEX RELAXATION

We would like to recover k-sparse $\mathbf{x}$ from measurements $\mathbf{b} = \mathbf{A}\mathbf{x}$ by solving the non-convex optimization problem:

$$\mathbf{x} = \operatorname*{arg\,min}_{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}} \|\mathbf{z}\|_0$$

Works if $\mathbf{A}$ has Kruskal rank $\geq 2k$, but very hard computationally.

Convex Relaxation: A very common technique. Just 'relax' the problem to be convex.

Basis Pursuit: $\mathbf{x} = \arg\min_{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}} \|\mathbf{z}\|_1$ where $\|\mathbf{z}\|_1 = \sum_{i=1}^{d} |\mathbf{z}(i)|$.

What is one algorithm we have learned for solving this problem?

- Projected (sub)gradient descent – convex objective function and convex constraint set.

We would like to recover k-sparse $\mathbf{x}$ from measurements $\mathbf{b} = \mathbf{A}\mathbf{x}$ by solving the non-convex optimization problem:

$$\mathbf{x} = \operatorname*{arg\,min}_{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}} \|\mathbf{z}\|_0$$

Works if $\mathbf{A}$ has Kruskal rank $\geq 2k$, but very hard computationally.

Convex Relaxation: A very common technique. Just 'relax' the problem to be convex.

Basis Pursuit: $\mathbf{x} = \arg\min_{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}} \|\mathbf{z}\|_1$ where $\|\mathbf{z}\|_1 = \sum_{i=1}^d |\mathbf{z}(i)|$.

What is one algorithm we have learned for solving this problem?

- Projected (sub)gradient descent – convex objective function and convex constraint set.
- An instance of linear programming, so typically faster to solve with a linear programming algorithm (e.g., simplex, interior point).

Why should we hope that the basis pursuit solution returns the unique k-sparse $\mathbf{x}$ with $\mathbf{A}\mathbf{x} = \mathbf{b}$? The minimizer $\mathbf{z}^*$ will have small ℓ_1 norm but why would it even be sparse?

$$\arg\min_{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}} \|\mathbf{z}\|_1 \quad \text{vs.} \quad \arg\min_{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}} \|\mathbf{z}\|_0$$

Why should we hope that the basis pursuit solution returns the unique k-sparse $\mathbf{x}$ with $\mathbf{A}\mathbf{x} = \mathbf{b}$? The minimizer $\mathbf{z}^*$ will have small ℓ_1 norm but why would it even be sparse?

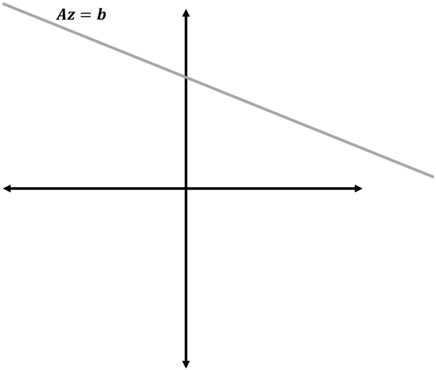
$$\underset{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}}{\arg\min} \|\mathbf{z}\|_1 \quad \text{vs.} \quad \underset{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}}{\arg\min} \|\mathbf{z}\|_0$$

Assume that $n = 1, d = 2, k = 1$. So $\mathbf{A} \in \mathbb{R}^{1\times 2}$ and $\mathbf{x} \in \mathbb{R}^2$ is 1-sparse.

Why should we hope that the basis pursuit solution returns the unique k-sparse $\mathbf{x}$ with $\mathbf{A}\mathbf{x} = \mathbf{b}$? The minimizer $\mathbf{z}^*$ will have small ℓ_1 norm but why would it even be sparse?

$$\arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \|\mathbf{z}\|_1 \quad \text{vs.} \quad \arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \|\mathbf{z}\|_0$$

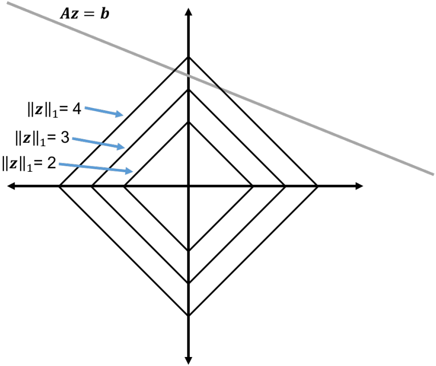
Assume that $n = 1, d = 2, k = 1$. So $\mathbf{A} \in \mathbb{R}^{1\times 2}$ and $\mathbf{x} \in \mathbb{R}^2$ is 1-sparse.



Why should we hope that the basis pursuit solution returns the unique k-sparse $\mathbf{x}$ with $\mathbf{A}\mathbf{x} = \mathbf{b}$? The minimizer $\mathbf{z}^*$ will have small ℓ_1 norm but why would it even be sparse?

$$\arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \|\mathbf{z}\|_1 \quad \text{vs.} \quad \arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \|\mathbf{z}\|_0$$

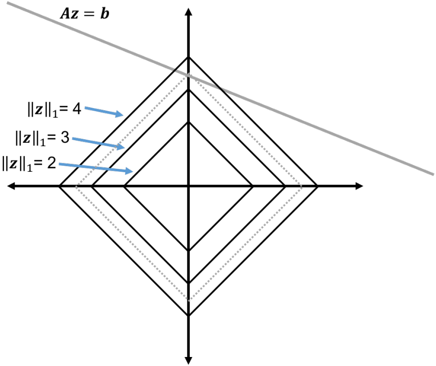
Assume that $n = 1, d = 2, k = 1$. So $\mathbf{A} \in \mathbb{R}^{1\times 2}$ and $\mathbf{x} \in \mathbb{R}^2$ is 1-sparse.



Why should we hope that the basis pursuit solution returns the unique k-sparse $\mathbf{x}$ with $\mathbf{A}\mathbf{x} = \mathbf{b}$? The minimizer $\mathbf{z}^*$ will have small ℓ_1 norm but why would it even be sparse?

$$\arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \|\mathbf{z}\|_1 \quad \text{vs.} \quad \arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \|\mathbf{z}\|_0$$

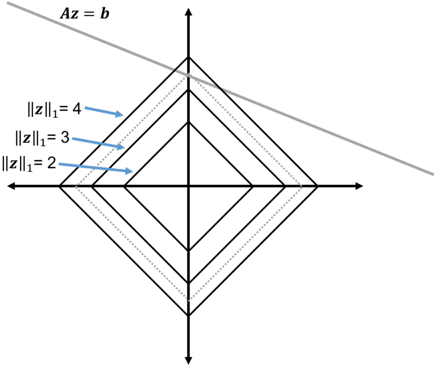
Assume that $n = 1, d = 2, k = 1$. So $\mathbf{A} \in \mathbb{R}^{1\times 2}$ and $\mathbf{x} \in \mathbb{R}^2$ is 1-sparse.



Why should we hope that the basis pursuit solution returns the unique k-sparse $\mathbf{x}$ with $\mathbf{A}\mathbf{x} = \mathbf{b}$? The minimizer $\mathbf{z}^*$ will have small ℓ_1 norm but why would it even be sparse?

$$\arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \|\mathbf{z}\|_1 \quad \text{vs.} \quad \arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \|\mathbf{z}\|_0$$

Assume that $n = 1, d = 2, k = 1$. So $\mathbf{A} \in \mathbb{R}^{1\times 2}$ and $\mathbf{x} \in \mathbb{R}^2$ is 1-sparse.

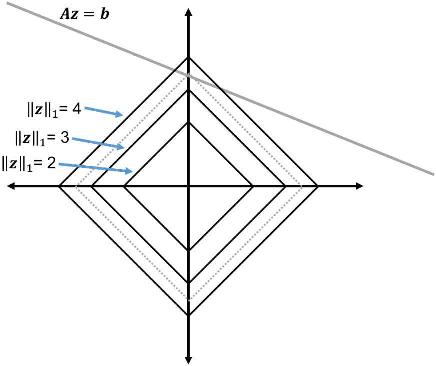


- Optimal solution will be on a corner (i.e., sparse), unless $\mathbf{A}\mathbf{z} = \mathbf{b}$ has slope 1.

Why should we hope that the basis pursuit solution returns the unique k-sparse $\mathbf{x}$ with $\mathbf{A}\mathbf{x} = \mathbf{b}$? The minimizer $\mathbf{z}^*$ will have small ℓ_1 norm but why would it even be sparse?

$$\underset{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}}{\arg\min} \|\mathbf{z}\|_1 \quad \text{vs.} \quad \underset{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}}{\arg\min} \|\mathbf{z}\|_0$$

Assume that $n = 1, d = 2, k = 1$. So $\mathbf{A} \in \mathbb{R}^{1 \times 2}$ and $\mathbf{x} \in \mathbb{R}^2$ is 1-sparse.

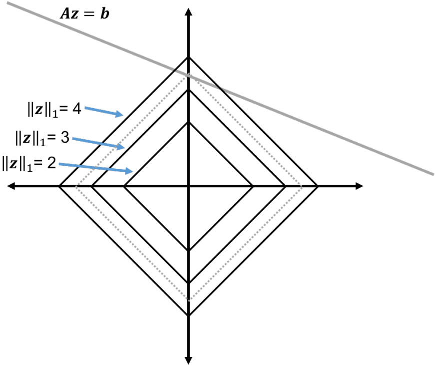


- Optimal solution will be on a corner (i.e., sparse), unless $\mathbf{A}\mathbf{z} = \mathbf{b}$ has slope 1.
- Similar intuition to the LASSO method.

Why should we hope that the basis pursuit solution returns the unique k-sparse $\mathbf{x}$ with $\mathbf{A}\mathbf{x} = \mathbf{b}$? The minimizer $\mathbf{z}^*$ will have small ℓ_1 norm but why would it even be sparse?

$$\arg\min_{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}} \|\mathbf{z}\|_2 \quad \text{vs.} \quad \arg\min_{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}} \|\mathbf{z}\|_0$$

Assume that $n = 1, d = 2, k = 1$. So $\mathbf{A} \in \mathbb{R}^{1 \times 2}$ and $\mathbf{x} \in \mathbb{R}^2$ is 1-sparse.

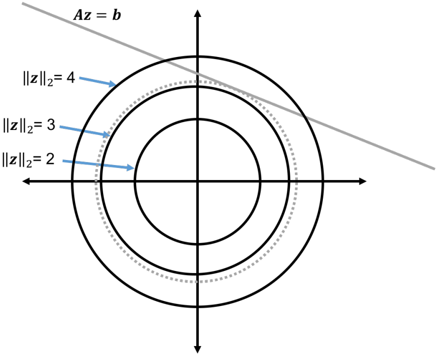


- Optimal solution will be on a corner (i.e., sparse), unless $\mathbf{A}\mathbf{z} = \mathbf{b}$ has slope 1.
- Similar intuition to the LASSO method.
- Does not hold if e.g., the ℓ_2 norm is used.

Why should we hope that the basis pursuit solution returns the unique k-sparse $\mathbf{x}$ with $\mathbf{A}\mathbf{x} = \mathbf{b}$? The minimizer $\mathbf{z}^*$ will have small ℓ_1 norm but why would it even be sparse?

$$\arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \qquad \text{vs.} \qquad \arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \|\mathbf{z}\|_0$$

Assume that $n = 1, d = 2, k = 1$. So $\mathbf{A} \in \mathbb{R}^{1\times 2}$ and $\mathbf{x} \in \mathbb{R}^2$ is 1-sparse.



- Optimal solution will be on a corner (i.e., sparse), unless $\mathbf{A}\mathbf{z} = \mathbf{b}$ has slope 1.
- Similar intuition to the LASSO method.
- Does not hold if e.g., the ℓ_2 norm is used.

BASIS PURSUIT THEOREM

Can prove that basis pursuit outputs the exact k-sparse solution $\mathbf{x}$ with $\mathbf{A}\mathbf{x} = \mathbf{b}$ (i.e, $\arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \|\mathbf{z}\|_1 = \arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \|\mathbf{z}\|_0$)

- Requires a strengthening of the Kruskal rank $\geq 2k$ assumption (that still holds in many applications).

Can prove that basis pursuit outputs the exact k-sparse solution $\mathbf{x}$ with $\mathbf{A}\mathbf{x} = \mathbf{b}$ (i.e, $\arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \|\mathbf{z}\|_1 = \arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \|\mathbf{z}\|_0$)

- Requires a strengthening of the Kruskal rank $\geq 2k$ assumption (that still holds in many applications).

Definition: $\mathbf{A} \in \mathbb{R}^{n\times d}$ has the (m,ϵ) restricted isometry property (is (m,ϵ)-RIP) if for all m-sparse vectors $\mathbf{x}$:

$$(1-\epsilon)\|\mathbf{x}\|_2 \leq \|\mathbf{A}\mathbf{x}\|_2 \leq (1+\epsilon)\|\mathbf{x}\|_2$$

Can prove that basis pursuit outputs the exact k-sparse solution $\mathbf{x}$ with $\mathbf{A}\mathbf{x} = \mathbf{b}$ (i.e, $\arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \|\mathbf{z}\|_1 = \arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \|\mathbf{z}\|_0$)

- Requires a strengthening of the Kruskal rank $\geq 2k$ assumption (that still holds in many applications).

Definition: $\mathbf{A} \in \mathbb{R}^{n\times d}$ has the (m,ϵ) restricted isometry property (is (m,ϵ)-RIP) if for all m-sparse vectors $\mathbf{x}$:

$$(1-\epsilon)\|\mathbf{x}\|_2 \leq \|\mathbf{A}\mathbf{x}\|_2 \leq (1+\epsilon)\|\mathbf{x}\|_2$$

Theorem: If $\mathbf{A}$ is $(3k,\epsilon)$-RIP for small enough constant ϵ, then $\mathbf{z}^\star = \arg\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{A}\mathbf{z}=\mathbf{b}} \|\mathbf{z}\|_1$ is equal to the unique k-sparse $\mathbf{x}$ with $\mathbf{A}\mathbf{x} = \mathbf{b}$ (i.e., basis pursuit solves the sparse recovery problem).

Wrap Up

Thanks for a great semester!

Randomization as a computational resource for massive datasets.

Randomization as a computational resource for massive datasets.

- Focus on problems that are easy on small datasets but hard at massive scale – set size estimation, load balancing, distinct elements counting (MinHash), checking set membership (Bloomfilters), frequent items counting (Count-min sketch), near neighbor search (locality sensitive hashing).

Randomization as a computational resource for massive datasets.

- Focus on problems that are easy on small datasets but hard at massive scale – set size estimation, load balancing, distinct elements counting (MinHash), checking set membership (Bloomfilters), frequent items counting (Count-min sketch), near neighbor search (locality sensitive hashing).
- Just the tip of the iceberg on randomized streaming/sketching/hashing algorithms.

Randomization as a computational resource for massive datasets.

- Focus on problems that are easy on small datasets but hard at massive scale – set size estimation, load balancing, distinct elements counting (MinHash), checking set membership (Bloomfilters), frequent items counting (Count-min sketch), near neighbor search (locality sensitive hashing).
- Just the tip of the iceberg on randomized streaming/sketching/hashing algorithms.
- In the process covered probability/statistics tools that are very useful beyond algorithm design: concentration inequalities, higher moment bounds, law of large numbers, central limit theorem, linearity of expectation and variance, union bound, median as a robust estimator.

Methods for working with (compressing) high-dimensional data

Methods for working with (compressing) high-dimensional data

- Started with randomized dimensionality reduction and the JL lemma: compression from *any* d-dimensions to $O(\log n/\epsilon^2)$ dimensions while preserving pairwise distances.

Methods for working with (compressing) high-dimensional data

- Started with randomized dimensionality reduction and the JL lemma: compression from *any* d-dimensions to $O(\log n/\epsilon^2)$ dimensions while preserving pairwise distances.
- Dimensionality reduction via low-rank approximation and optimal solution with PCA/eigendecomposition/SVD.

Methods for working with (compressing) high-dimensional data

- Started with randomized dimensionality reduction and the JL lemma: compression from *any* d-dimensions to $O(\log n/\epsilon^2)$ dimensions while preserving pairwise distances.
- Dimensionality reduction via low-rank approximation and optimal solution with PCA/eigendecomposition/SVD.
- Low-rank approximation of similarity matrices and entity embeddings (e.g., LSA, word2vec, DeepWalk).

Methods for working with (compressing) high-dimensional data

- Started with randomized dimensionality reduction and the JL lemma: compression from *any* d-dimensions to $O(\log n/\epsilon^2)$ dimensions while preserving pairwise distances.
- Dimensionality reduction via low-rank approximation and optimal solution with PCA/eigendecomposition/SVD.
- Low-rank approximation of similarity matrices and entity embeddings (e.g., LSA, word2vec, DeepWalk).
- Low-rank structure in graphs – nonlinear dimensionality reduction and spectral clustering for community detection, stochastic block model, matrix concentration.

Methods for working with (compressing) high-dimensional data

- Started with randomized dimensionality reduction and the JL lemma: compression from *any* d-dimensions to $O(\log n/\epsilon^2)$ dimensions while preserving pairwise distances.
- Dimensionality reduction via low-rank approximation and optimal solution with PCA/eigendecomposition/SVD.
- Low-rank approximation of similarity matrices and entity embeddings (e.g., LSA, word2vec, DeepWalk).
- Low-rank structure in graphs – nonlinear dimensionality reduction and spectral clustering for community detection, stochastic block model, matrix concentration.
- In the process covered linear algebraic tools that are very broadly useful in ML and data science: eigendecomposition, singular value decomposition, projection, norm transformations.

Foundations of continuous optimization and gradient descent.

Foundations of continuous optimization and gradient descent.

- Motivation for continuous optimization as loss minimization in ML. Foundational concepts like convexity, convex sets, Lipschitzness, directional derivative/gradient.

Foundations of continuous optimization and gradient descent.

- Motivation for continuous optimization as loss minimization in ML. Foundational concepts like convexity, convex sets, Lipschitzness, directional derivative/gradient.
- How to analyze gradient descent in a simple setting.

Foundations of continuous optimization and gradient descent.

- Motivation for continuous optimization as loss minimization in ML. Foundational concepts like convexity, convex sets, Lipschitzness, directional derivative/gradient.
- How to analyze gradient descent in a simple setting.
- Online optimization, online gradient descent, and how to use it to analyze stochastic gradient descent (by far the most common optimization method in ML).

Foundations of continuous optimization and gradient descent.

- Motivation for continuous optimization as loss minimization in ML. Foundational concepts like convexity, convex sets, Lipschitzness, directional derivative/gradient.
- How to analyze gradient descent in a simple setting.
- Online optimization, online gradient descent, and how to use it to analyze stochastic gradient descent (by far the most common optimization method in ML).
- Lots that we didn't cover: accelerated methods, adaptive methods, second order methods (quasi-Newton methods), practical considerations. Hopefully gave mathematical tools to understand these methods.

- The weirdness of high-dimensional space and geometry. Connections to randomized methods, dimensionality reduction. Always useful to keep in mind.
- Compressed sensing/sparse recovery – a very broad and widely-used framework for working with high-dimensional data. Connection to streaming algorithms (frequent items counting) and convex optimization.

Thanks!